Databases -2- Practical Labs

3th Year





Lab -1- Introduction to PL/SQL

Introduced By:

Eng. Khaled Kondakji   *Eng. Ibrahim Ayoub*

Supervised By:

Dr. Eng. Asmaa Shaar

Introduction to PL/SQL

مقدمة:

- هي لغة برمجية إجرائية (Procedural Language) امتداداً للغة SQL تساعد في تجاوز بعد القيود الموجودة في لغة SQL، مطورة من شركة Oracle.
- تتكون هذه اللغة بشكل أساسي من مجموعة من كتل برمجية (Blocks)، والذي يحوي كل منها مجموعة الأقسام التالية:
  - ✓ **الرأس (Header):** يستخدم للكتل التي تتضمن تعريف توابع، إجرائيات وغيرها...
  - ✓ **التصريح (Declaration):** يستخدم للتصريح عن المتحولات والمؤشرات والاستثناءات.
  - ✓ **التنفيذ (Execution):** نضع هنا تعليمات البرمجية بلغة PL/SQL والتي تؤدي وظيفة معينة.
  - ✓ **الاستثناءات (Exceptions):** يستخدم هذا القسم لمعالجة الاستثناءات والأخطاء.
- يجب أن يحوي أي برنامج على قسم التنفيذ بشكل إجباري فيما تكون باقي الأقسام اختيارية عند الضرورة،
- يمكن أن تكون هذه الكتل (anonymous) لا تحوي على اسم وتبدأ بإحدى التعليمتين (declare or begin)، ولا يمكن استدعاؤه بواسطة كتل أخرى. ويمكن أن تحوي اسم وتستخدم هذه الكتل من أجل تعريف التوابع والإجرائيات.

تعريف anonymous blocks:

- كما ذكرنا سابقاً تبدأ هذه الكتل بإحدى التعليمتين (declare or begin) ولا تحتوي على ترويسة (header).
- تحوي هذه الكتل تعليمات برمجية مكتوبة بلغة PL/SQL والشكل العام لهذا النوع من الكتل كما يلي:

```
[ DECLARE ... declaration statements ... ] BEGIN
... one or more executable statements ...
[ EXCEPTION
... exception handler statements ... ]
END;
```

حيث تشير الأقواس المربعة لقسم اختياري ويجب أن يحوي البرنامج على الأقل على تعليميتي (begin and end) وعلى الأقل أمر قابل للتنفيذ. أما الآن سنقوم بتنفيذ المثال التالي:

```
DECLARE
    V NUMBER;
BEGIN
    V := 5;
    DBMS_OUTPUT.PUT_LINE('hello world');
    DBMS_OUTPUT.PUT_LINE(V);
END;
```

ملاحظة: لتنفيذ البرنامج السابق نختار من القائمة (view) إظهار قسم (DBMS output) ومن ثم نضغط زر تنفيذ البرنامج ويكون ناتج التنفيذ كما يلي:

```
hello world
5

PL/SQL procedure successfully completed.

Commit complete.
```

في المثال السابق قمنا بالتصريح عن متحول (v) من نمط (number) و عند التنفيذ تعليمة الطباعة الأولى تقوم بطباعة عبارة الترحيب والأخرى تطبع قيمة المتحول (v).

التصريح عن المتحولات:

- يتم تعريف عن المتحولات ضمن قسم التصريح (Declaration).
- ويتم استخدامها واسناد القيم لها ضمن قسم التنفيذ (Execution)، كما ويمكننا أن نمرر قيمها للإجرائيات والتوابع وطباعتها باستخدام تعليمات الطباعة.
- الشكل العام للتصريح عن المتحولات:

```
variable_name [CONSTANT] datatype [NOT NULL] [ := | DEFAULT init_value];
```

- يتم تهيئة المتحولات إما باستخدام (=:) أو قيمة افتراضية عن طريق (DEFAULT).
- عند استخدام القيد (NOT NULL) يجب أن يحوي المتحول على قيمة ابتدائية.

الآن سنقوم بتنفيذ المثال التالي والذي يوضح لنا آلية التعامل مع المتحولات بمختلف أنماطها:

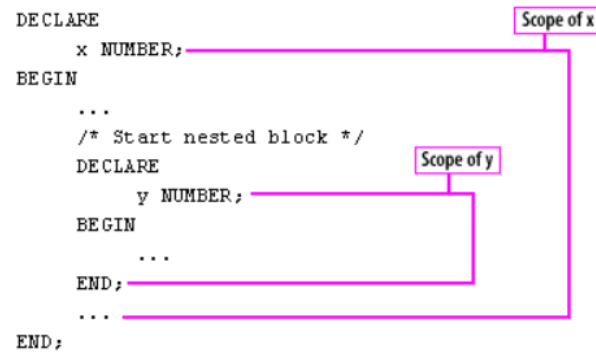
```
DECLARE
 -- define dates variables:
    V_DATE       DATE;
    TODAY        DATE:=SYSDATE;
 -- define numbers variables:
    NO           NUMBER:=10;
    MUL          NUMBER:=10*2;
    SALARY       NUMBER(9, 2);
 -- define string variables:
    F_NAME       VARCHAR(100) NOT NULL:='khaled';
    L_NAME       NVARCHAR2(100) DEFAULT 'Unknown';
    ENOUGH_MONEY BOOLEAN; -- define boolean variable.
    V_PI         CONSTANT NUMBER:=3.14; --define constant variable.
BEGIN
    v_date := '10-May-2012';
    DBMS_OUTPUT.PUT_LINE(V_DATE);
    DBMS_OUTPUT.PUT_LINE(TODAY);
    DBMS_OUTPUT.PUT_LINE(NO);
    SALARY := 105 * 25.6;
    DBMS_OUTPUT.PUT_LINE(SALARY);
    DBMS_OUTPUT.PUT_LINE(F_NAME);
    F_NAME := 'Ibrahim';
    DBMS_OUTPUT.PUT_LINE(F_NAME);
 -- v_pi = 10; -- this will cause error !!
END;
```

أما نتيجة تنفيذ المثال كما يلي:

```
10-MAY-12
22-MAR-24
10
2688
khaled
Ibrahim
```

مجالات الرؤيا (Visibility):

- هو عبارة عن جزء من البرنامج (كتل أو توابع أو..) حيث يمكننا الوصول للمتحولات ضمنها.
- لا يمكننا الوصول لأي متحول معرف ضمن كتلة داخلية من أي كتلة خارجية تحويها.
- باستخدام الحزم (packages) يمكننا إنشاء متحولات global يمكن الوصول إليها من أي كتلة من البرنامج.



أما الآن سنقوم بتنفيذ الأمثلة التالية مع ملاحظة آلية الوصول للمتحولات بحسب نوعها (local or global).

المثال الأول:

```
DECLARE
    V_GLOBAL VARCHAR2(100):= 'This is a global variable';
BEGIN
    DECLARE
        V_LOCAL VARCHAR2(100):='This is local variable';
    BEGIN
        DBMS_OUTPUT.PUT_LINE(V_GLOBAL);
        DBMS_OUTPUT.PUT_LINE(V_LOCAL);
    END;
    DBMS_OUTPUT.PUT_LINE(V_GLOBAL);
 -- DBMS_OUTPUT.PUT_LINE(v_local); -- this will cause error !!
END;
```

وتكون نتيجة تنفيذ المثال كما يلي:

```
This is a global variable
This is local variable
This is a global variable
```

المثال الثاني:

```
DECLARE
    FATH_NAME VARCHAR2(100):='Tom';
    BIRTH_DAY DATE:='20-Jul-1981';
BEGIN
    DECLARE
        CHILD_NAME VARCHAR2(100):='Layal';
        BIRTH_DAY  DATE:='5-Apr-2013';
    BEGIN
        DBMS_OUTPUT.PUT_LINE('the father name is ' || FATH_NAME);
        DBMS_OUTPUT.PUT_LINE('the father birthday is ' ||
BIRTH_DAY);
        DBMS_OUTPUT.PUT_LINE('the child name is ' || CHILD_NAME);
        DBMS_OUTPUT.PUT_LINE('the child birthday is '|| BIRTH_DAY);
    END;
END;
```

وتكون نتيجة تنفيذ المثال كما في الشكل التالي:

```
the father name is Tom
the father birthday is 05-APR-13
the child name is Layal
the child birthday is 05-APR-13
```

نلاحظ أنه لا يمكننا الوصول لقيمة عيد ميلاد الأب بسبب استخدامنا نفس اسم المتحول ضمن الكتلة الداخلية ولحل هذه المشكلة يمكننا استخدام مفهوم التسمية (Label) والتي تستخدم لتسمية كتلة موجودة لدينا، الآن سنقوم بتعديل المثال السابق من خلال استخدام هذه الميزة ونلاحظ تنفيذ البرنامج السابق:

```
<<outerblock>>
DECLARE
    FATH_NAME VARCHAR2(100):='Tom';
    BIRTH_DAY DATE:='20-Jul-1981';
BEGIN
    DECLARE
        CHILD_NAME VARCHAR2(100):='Layal';
        BIRTH_DAY  DATE:='5-Apr-2013';
    BEGIN
        DBMS_OUTPUT.PUT_LINE('the father name is ' || FATH_NAME);
        DBMS_OUTPUT.PUT_LINE('the father birthday is ' ||
outerblock.BIRTH_DAY);
        DBMS_OUTPUT.PUT_LINE('the child name is ' || CHILD_NAME);
        DBMS_OUTPUT.PUT_LINE('the child birthday is '|| BIRTH_DAY);
    END;
END;
```

باستخدام الاسم الخاص بالكتلة الخارجية بات بإمكاننا التمييز بين قيمتي عيد ميلاد الأب والابن من خلال الوصول لقيمة عيد ميلاد الأب عن طريق هذه الاسم الخاص. وبالتالي يكون تنفيذ البرنامج السابق كما يلي:

```
the father name is Tom
the father birthday is 20-JUL-81
the child name is Layal
the child birthday is 05-APR-13
```

ربط نتيجة الاستعلامات بالمتحولات (bind variables):

- يتم إنشاء هذه المتحولات باستخدام الكلمة المفتاحية (Variable)، ويمكننا الوصول إليها حتى بعد الانتهاء من تنفيذ البرنامج من خلال استخدام هذه الإشارة (:) قبل اسم المتحول.
- سنقوم الآن بكتابة برنامج يستعلم من جدول الموظفين (employees)، وهذا الجدول يمكننا الوصول إليه بمجرد تفعيل حساب مستخدم (HR) الذي قمنا بتفعيله في الجلسة السابقة.
- أولاً قمنا بتعديل المستخدم الذي ننفذ البرنامج منه ليصبح المستخدم (HR) بدلاً من المستخدم (SYS) وذلك بتنفيذ التعليمة التالية:

```
-- this statement must be executed before run the code:
ALTER SESSION SET current_schema = HR;
---------------------------------------------------------------
-- Now we're executing as HR user instead os sys user.
```

وبالتالي الآن يمكننا الوصول للجداول الموجودة ضمن HR.

```
SQL> -- this statement must be executed before run the code:
SQL> ALTER SESSION SET current_schema = HR;

SESSION altered.

SQL> ---------------------------------------------------------------
SQL> -- Now we're executing as HR user instead os sys user.
SQL> DECLARE
```

- أما البرنامج الذي سنقوم بتنفيذه سيقوم بجلب معلومات الموظف ذو الرقم (100) من جدول الموظفين وتخزين هذه البيانات ضمن المتحولات التي قمنا بتعريفها مسبقاً.

ملاحظة: نستخدم رمز (%) والتي تسمح لنا بإسناد نفس نمط حقل الجدول أو متحول معرف مسبقاً للمتحول الحالي الذي نقوم بالتصريح عنه.

```
DECLARE
    V_FIRST     EMPLOYEES.FIRST_NAME%TYPE;
    V_LAST      EMPLOYEES.LAST_NAME%TYPE;
    V_SAL       NUMBER;
    "hire date" DATE; -- this is not recommended from oracle
    BEGIN
    SELECT
        FIRST_NAME, LAST_NAME, SALARY, HIRE_DATE
        INTO V_FIRST, V_LAST, V_SAL, "hire date"
    FROM EMPLOYEES
    WHERE EMPLOYEE_ID = 100;
    DBMS_OUTPUT .PUT_LINE('the employee first name is '|| V_FIRST);
    DBMS_OUTPUT .PUT_LINE('the employee last name is ' || V_LAST);
    DBMS_OUTPUT .PUT_LINE('the employee salary is ' || V_SAL);
    DBMS_OUTPUT .PUT_LINE('the employee hire date is ' || "hire
date");
    DBMS_OUTPUT .PUT_LINE('the employee first name length is ' ||
LENGTH(V_FIRST) );
    DBMS_OUTPUT .PUT_LINE('the first 3 letters of first name is ' ||
SUBSTR(V_FIRST, 1, 3));
END;
```

ويكون لدينا تنفيذ البرنامج السابق كما يلي:

```
the employee first name is Steven
the employee last name is King
the employee salary is 24000
the employee hire date is 17-JUN-03
the employee first name length is 6
the first 3 letters of first name is Ste
```

معالجة الاستثناءات (Exception Handling):

- تشير هذه الاستثناءات لأخطاء وقعت أثناء تنفيذ البرنامج ويمكن أن تحدث من قبل النظام مثلاً بسبب امتلاء الذاكرة أو تكرار في قيم الأدلة، أو نتيجة تنفيذ خاطئ من قبل المستخدم.
- يمكن معالجة هذه الاستثناءات من خلال إضافة قسم (Exception) وتحديد الآلية التي يمكننا معالجة الاستثناء الحاصل. كما في المخطط التالي:

```
DECLARE
   exception_name EXCEPTION;
BEGIN
   …
   IF CONDITION
   THEN
      RAISE exception_name;
   END IF;
   …
EXCEPTION
   WHEN exception_name
   THEN
      ERROR-PROCESSING STATEMENTS;
END;
```

سنستخدم هذه الآلية كما في المثال التالي:

```
-- this statement must be executed before run the code:
ALTER SESSION SET CURRENT_SCHEMA = HR;
---------------------------------------------------------------
-- Now we're executing as HR user instead os sys user.
DECLARE
    V_EMP_ID     EMPLOYEES.EMPLOYEE_ID%TYPE :=99;
    V_SAL        EMPLOYEES.SALARY%TYPE;
    E_INVALID_ID EXCEPTION;
BEGIN
    IF V_EMP_ID < 100 THEN
        RAISE E_INVALID_ID;
    END IF;
    SELECT SALARY INTO V_SAL
    FROM EMPLOYEES
    WHERE EMPLOYEE_ID = V_EMP_ID;
    DBMS_OUTPUT.PUT_LINE('The salary is ' || V_SAL);
    DBMS_OUTPUT.PUT_LINE('no exception has been raised');
EXCEPTION
    WHEN E_INVALID_ID THEN
        DBMS_OUTPUT.PUT_LINE('invalid employee id');
END;
```

- نلاحظ أننا قمنا باستخدام شرط (if) والذي يختبر قيمة معرف الموظف ويقوم بإخطارنا في حالة كانت قيمة المعرف أقل من (100) من خلال استخدام التعليمة (raise).
- بمجرد حدوث الإخطار يتم نقل التنفيذ لقسم معالجة الخطأ دون متابعة تنفيذ البرنامج بالشكل الصحيح كما في نتيجة تنفيذ المثال السابق:

```
invalid employee id

PL/SQL procedure successfully completed.

Commit complete.
```

ملاحظة قم بتجربة تنفيذ البرنامج من أجل معرف موظف (101).

```
The salary is 17000
no exception has been raised

PL/SQL procedure successfully completed.

Commit complete.
```